WissGrid

Deliverable 2.1.6

Arbeitspaket 2: Blaupausen und Beratung

Leitfaden zur Unterstützung des Software-Management-Prozesses[1]

Deliverable	2.1.6 Leitfaden Software-Management-Prozesse
Autoren	Arbeitspaket 2: Blaupausen und Beratung
Editoren	C. Grimme
Datum	12-04-2011
Dokument Version	1.0.2

A: Status des Dokuments

Deliverable 2.1.6, Version 1.0.2, 2011-Iteration mit geringen Änderungen zu Vorversion, potentiell RFC.

B: Bezug zum Projektplan

Dieser Leitfaden stellt ein generisches Modell für die Etablierung eines Software-Managementprozesses innerhalb von Community-Grids zur Verfügung.

C: Abstract

Der Leitfaden entwickelt einen modularen Software-Managementprozess, der speziell auf Bedürfnisse und Rahmenbedingungen von Community-Grid-Projekten ausgerichtet ist. Das Dokument basiert

[1]This work is created by the WissGrid project. The project is funded by the German Federal Ministry of Education and Research (BMBF).

dabei auf Erfahrungen aus der Materialsammlung D 2.1.1. Es definiert einen grundlegenden Prozess, der mit optionalen Komponenten (je nach Anforderung der umsetzenden Projekte) erweitert werden kann.

D: Änderungen

Version	Date	Name	Brief summary
0.1.0	11.11.2009	C. Grimme	Erstellung des Arbeitsdokumentes
0.1.1	16.11.2009	C. Grimme	Erste Struktur
0.1.2	18.11.2009	C. Grimme	Einleitung und Anpassung der Struktur
0.1.3	23.11.2009	C. Grimme	Zusammenfassung Materialsammlung (bzgl. SLM)
0.1.4	07.12.2009	C. Grimme	Überarbeitung der Materialauswertung
0.1.5	14.12.2009	C. Grimme	Materialauswertung und Struktur
0.1.6	05.01.2009	C. Grimme	Edition Rahmenbedingungen abgeschlossen
0.1.7	08.01.2009	C. Grimme	Grafischer Prozessentwurf und Struktur
0.1.8	13.01.2009	C. Grimme	Beschreibung des Prozesses, Elemente
0.1.9	14.01.2009	C. Grimme	Vervollständigung Rollen
0.2.0	15.01.2009	C. Grimme	Fertigstellung Draft
0.2.1	28.01.2009	T. Rathmann	Korrekturen
0.2.1	02.02.2009	F. Schlünzen	Korrekturen
0.2.1	02.02.2009	H. Enke	Korrekturen
0.2.1	17.02.2009	C. Grimme	Zielsetzung hinzugefügt
1.0.0	23.03.2010	C. Grimme	Release, finale Version
1.0.1	04.03.2011	C. Grimme	Update und Kommentare
1.0.2	12.04.2011	C. Grimme	Update, Potentiell RFC

E:

Zielgruppen des Dokuments

Dieses Dokument stellt in der vorliegenden Version einen konzeptionellen Entwurf für das Softwaremanagement im Grid dar. Es richtet sich damit insbesondere an Entscheidungsträger innerhalb einer Wissenschafts-Community und liefert erste Anhaltspunkte für den Ablauf und die Organisation eines Softwareentwicklungsprozesses. Dabei werden folgende Aspekte im Dokument angesprochen:

Evaluation von Softwareentwicklung im Grid: Basierend auf Erhebungen zum Softwaremanagement in existierenden Grid-Projekten werden verschiedene Voraussetzungen für die Softwareentwicklung im Grid-Kontext untersucht und Rahmenbedingungen festgelegt. Dies soll insbesondere Entscheidungsträgern (Projektinitiator, Projektleitung) Anhaltspunkte für eine realistische Planung der Softwareentwicklung im Grid ermöglichen. Gleichzeitig soll dies Erfahrungen etablierter Projekte transportieren und die Verantwortlichen neuer Communitiy-Grids für formale, organisatorische und zeitliche Rahmenbedingungen sensibilisieren. Abschließend eignet sich diese erste Version des Dokumentes zur Motivation eines an den Grid-Kontext angepassten Softwaremanagementprozess der bisherige Best-Practices einbezieht und trotzdem formale Regeln für eine geordnete Entwicklung aufstellt.

Darstellung der Prozessidee: Die Prozessidee für die Entwicklung im Grid-Kontext richtet sich an alle Beteiligten eines Community-Grids. Hier wird der Gedanke des Meilenstein-getriebenen evolutionären Prozesses grundlegend dargestellt und aus den zuvor diskutierten Voraussetzungen motiviert. Diese Idee ist sowohl an Projektverantwortliche, Techniker wie auch Nutzer gerichtet und stellt einen einfachen Einstieg in ein iteratives Vorgehen dar.

Generelle Prozessspezifizierung: Im Dritten Teil des Dokumentes wird der zuvor beschriebene Softwareprozess auf oberster Ebene allgemein spezifiziert. Dies beinhaltet einerseits die detaillierte Darstellung von Rollen, Artefakten (also Ergebnissen und Gegenständen des Prozesse) sowie Aktivitäten, andererseits die zeitliche Verknüpfung dieser Elemente. Diese erste allgemeine Darstellung richtet sich insbesondere an Projektverantwortliche, die für die Etablierung, Umsetzung und Einhaltung des Prozesses Verantwortlich sind. Die hier vorgestellten Konzepte stellen ein Rahmenwerk für eine spezifische Instanziierung des Softwaremanagementprozesses im Kontext eines Community-Grids dar.

Inhaltsverzeichnis

Summary

This document provides a blueprint for a Grid-specific software development process starting with a revision of experiences from completed development projects in the Grid context. Based on that, specific environmental conditions for such a process are analyzed to give a reasonable framework for process conception. Eventually, an evolutionary milestone-driven software process is defined, basic definitions and notations are given, and the upper level of roles, activites, artefacts and workflows is specified.

How to read the document:

For a high-level view:	digest the introduction to each chapter, the subsections of chapter 1, and the summaries of each other chapter.
For evaluation of the community developments:	chapter 1+2,
For in depth study and technical details:	all of the document, especially chapter 3

Einleitung

Die D-Grid-Initiative verschiedener Fachwissenschaftler, Forschungseinrichtungen und Techniker ermöglichte in ihrer ersten Phase die Etablierung einer vielfältigen nationalen Grid-Infrastruktur. Im Gegensatz zu vielen internationalen und fachwissenschaftlich übergreifenden Infrastrukturentwicklungen wurden in Deutschland bewusst spezielle technische Lösungen für die einzelnen akademischen Fachnutzer unterstützt und somit eine vielfältige Landschaft technischer Lösungsansätze und Realisierungen geschaffen. Dies führte neben den unterschiedlichen Ausprägungen der jeweiligen Grid-Infrastruktur zu recht diversen Ansätzen für Organisationsstruktur und Arbeitsorganisation, insbesondere auch bei dem Prozess der Softwareentwicklung und der Produktpflege. Einerseits bringt die Innovation des Grids technische und infrastrukturelle, aber auch konzeptionelle und organisatorische Neuerungen mit sich, die einen starken Einfluss auf das Software-Management haben. Andererseits definieren die Förderrichtlinien des BMBF verschiedene Randbedingungen, die in Prozesse und Organisation einzufließen müssen. Gleichzeitig hat die fachliche Einbindung einiger Projekte in übergreifende Forschungsvorhaben großen Einfluss auf das durchgeführte Software-Management. Daraus ergibt sich eine Vielzahl von Vorgehensmodellen, Techniken und Organisationsformen, die bereits in einem ersten Arbeitsschritt im WissGrid-Projekt untersucht wurden (Deliverable 2.1.1, Materialsammlung).

Das vorliegende Dokument entwickelt nun ausgehend von der Materialsammlung aus den unterschiedlichen Ansätzen, Vorgehensweisen und Erfahrungen einen Leitfaden für die Etablierung eines angemessenen Software-Managements in akademischen Community-Grid-Projekten.

Dabei wird zuerst eine ausführliche Auswertung der bzgl. des Software-Managements gesammelten Materialien aus dem vorangegangenen Deliverable 2.1.1 vorgenommen und wichtige Aspekte kondensiert dargestellt und diskutiert. Sie werden in den Kontext etablierter Software-Management-Prozesse aus der Literatur gestellt und Übereinstimmungen und Unterschiede herausgearbeitet. Anschließend wird ein umfassendes, vollständig konfigurierbares Software-Management-Modell für die Entwicklung in Community-Grids vorgestellt und erläutert. Dabei wird einerseits das allgemeine Konzept vorgestellt, andererseits auf die einzelnen ,,Teilkomponenten" eingegangen. Hier stehen insbesondere organisatorische, zeitliche sowie technische Komponenten im Mittelpunkt.

Kapitel 1

Auswertung der Materialsammlung zum Software-Managementprozess

Das vorliegende Kapitel fasst die in Deliverable 2.1.1 gesammelten Ergebnisse der Erhebung zu Software-Managementprozessen in Community-Grids zusammen und dient damit als eine Art Anforderungskatalog und Grundlage für die Entwicklung einer Blaupause für Software-Managementprozesse in Grids. Die Zusammenfassung setzt sich aus Aspekten zusammen, die von verschiedenen Community-Grids als Vorgehensweise beschrieben und als perspektivisch wünschenswert eingestuft wurden. Strukturell richtet sich dieses Kapitel stark nach den einzelnen Aspekten der Materialerhebung.

1.1 Ermittlung und Umgang mit Anforderungen

Dieser Abschnitt fasst die bisherigen Vorgehensweisen der Community-Grids bei der Anforderungserhebung und -analyse zusammen. Abschließend werden Meinungen und Anregungen zur Verbesserung dargelegt.

1.1.1 Vorgehen zur Anforderungsermittlung

Bei der Ermittlung der Anforderungen sind für akademische Communities bisher zwei verschiedene Zielsetzungen und damit verbundene Ansätze zu identifizieren, die stark von dem Kontext abhängig sind, in dem ein Community-Grid angesiedelt ist:

Etablierung eines Community-Grids: Dieses Ziel der Etablierung eines Community-Grids innerhalb einer akademischen Disziplin setzt für die Anforderungsermittlung eine starke Beteiligung der Nutzer voraus. Die meisten akademischen Projekte im D-Grid integrierten aus diesem Grund die Nutzer in die Erfassung der Anforderungen. Neben der direkten Beteiligung von Nutzern im Projekt wurden Wissenschaftler unterschiedlicher Einrichtungen und Institute gezielt angesprochen. Dabei wurden vielfach vorbereitende Workshops ausgerichtet sowie anhand von Fragebögen die Interessen der Nutzer gesammelt.

Integration in eine Projektstruktur: Bei diesem Ansatz gliederte sich ein Projekt in eine existierende und übergreifende Projektstruktur ein. Damit geben die übergeordneten Projektziele

und Rahmenbedingungen einen Großteil der Anforderungen an zu entwickelnde Software vor. Aus den akademischen Projekten im D-Grid ist hier das HEP-CG zu nennen, das sich von Beginn an in die existierende CERN-Infrastruktur eingliederte und die Anforderungen für Entwicklungen anhand definierter Vorgaben bezüglich einzelner Experimente am Kernforschungszentrum ermitteln konnte. Dabei waren Nutzer meist zugleich an der Umsetzung beteiligte Techniker.

1.1.2 Prozess der Anforderungsanalyse:

Der Prozess der Anforderungsanalyse stellte sich in vielen Projekten unterschiedlich dar. Dieser kurze Abschnitt versucht alle relevanten zeitlichen wie inhaltlichen Komponenten der Anforderungsanalyse zu sammeln.

Zeitliche Aspekte

- Die Antragsphase ermöglichte eine erste Analyse von Anforderungen innerhalb der Community und verlangte eine grobe Planung von Arbeitsschritten sowie teilweise einen ersten Entwurf einer Architektur.
- Zu einem frühen Zeitpunkt wurden oft Anforderungen aus den bisherigen Anwendungserfahrungen von Fachwissenschaftlern in die Anforderungsanalyse integriert.
- Am Projektbeginn stand zumeist eine ausführliche Evaluationsphase zur konkreteren Bestimmung von Anforderungen (Use Cases einzelner Nutzergruppen) als Voraussetzung für den ersten Entwurf und eine prototypische Implementierung.
- Nachfolgend wurden Analysen aus kontinuierlichen Anforderungserhebungen auf Workshops (Nutzer und Techniker) integriert.

Inhaltliche Aspekte

- Bei den meisten Projekten standen zuerst funktionale Anforderungen im Vordergrund, die sich direkt aus der Anforderungsanalyse ergaben. Diese Anforderungen waren jedoch meist nicht statisch, sondern veränderten sich im Laufe des Projektes (auch aufgrund von vorherigen Kommunikationsproblemen) oder wurden aufgrund des Nutzer-Feedbacks angepasst.
- Im späteren Projektverlauf traten nicht-funktionale Anforderungen als gleichwertig oder wichtigere Aspekte auf. Darunter fielen folgende Punkte:
  - Handhabbarkeit der entwickelten Software
  - Einbettung in ,,natürliche" Nutzungsumgebung des Nutzers (z.B. Einbettung in die Eclipse-Umgebung der Anwendungen bei TextGrid, um eine gewohnte Anwendungsumgebung zu garantieren.)
  - Zuverlässigkeit der Software im Produktiveinsatz
  - Sicherheit
  - Performance

  - Erweiterbarkeit
  - Nachhaltigkeit (Verfügbarkeit und Pflegbarkeit)

1.1.3 Produkte der Analyse

Als Produkte der Anforderungsanalyse wurden die folgenden Dokumente, Seiten und Spezifikationen genannt:

- Dokumentation von Anforderungen in Form von:
  - Verschiedenen Diagrammen: Ablaufdiagramme, Strukturdiagramme, Flussdiagramme
  - Online-Beschreibungen: Ablage von Analyseergebnissen im Projekt-Wiki oder auf der Projekt-Webseite
  - Nutzungsszenarien, Workflows und Nutzerbeschreibungen
- Projektantrag als Resultat der zeitlich ersten Analysephase. Hier sind bereits einige Rahmenbedingungen und Anwendungsfälle festgelegt.
- Schnittstellendefinitionen aufbauend auf detaillierter Analyse und ersten Entwürfen.
- Prototypische Demonstratoren, um die funktionalen Anforderungen während der ersten Planung und der Entwicklung überprüfbar zu machen.
- Toolspezifikation, technische Entscheidungen über den Einsatz von technischen Hilfsmitteln, Entwicklungsumgebungen und Softwareinfrastruktur (z.B. Grid-Middleware, WebServices usw.)

1.1.4 Zusammenfassung und Meinungen:

Ein Großteil der Community-Grids sieht einen starken Bedarf für einen wohldefinierten Prozess zur Ermittlung, Aufnahme, Analyse und Dokumentation von Anforderungen.

Insbesondere sind Mittel zu entwickeln, um die Kommunikationshürde zwischen Fachwissenschaftlern in der Rolle der Nutzer und den Technikern zu überwinden. Dazu können nach Ansicht der Befragten mehrere Ansätze genutzt werden:

- Vorsehen von Vermittlungsstellen zwischen den beiden Rollen der Nutzer und Techniker im Prozess der Anforderungsanalyse.
- Ausführliche und gemeinsame Diskussion sowie Dokumentation der Anforderungen. Dazu können ggf. verstärkt gemeinsame Veranstaltungen oder Arbeitsgruppen genutzt werden.
- Festlegen eines Prozesses zur Aufbereitung der Anforderungsanalyse, dessen Ergebnis von Nutzern und Technikern gleichermaßen verwendet werden kann.

1.2 Organisationsstrukturen und Vorgehensmodelle

Die innere Organisationsstruktur des Projektes, der Arbeitsgruppen und Projektteams ist ebenso wie das oft daraus resultierende Vorgehensmodell der Softwareentwicklung Gegenstand dieses Abschnittes. Abschließend werden wieder Meinungen und Verbesserungsvorschläge zusammengefasst.

1.2.1 Organisation des Entwicklungsprozesses:

Die Untersuchung der Organisation von Softwareentwicklung lässt sich im Kontext der akademischen Community-Grids auf drei verschiedenen Ebenen untersuchen. Eine entsprechende Betrachtungsweise ergibt sich leicht aus der Konzeption der meisten Projekte. Als einzige Ausnahme ist das HEP-Projekt zu nennen, das sich im LHC-Umfeld an den Vorgaben des WLCG orientiert. Die speziell für die deutsche HEP-Community eingerichtete Analysis Facility (NAF) richtet neben dem gesamten Software-Managementprozess auch seine Organisationsstruktur nach den Vorgaben der TeraScale Allianz aus.

Die oberste Ebene beschreibt die Organisation des Gesamtprojektes, die Rollen und das gesamte zeitliche Management des Projektes. Die zweite Ebene betrachtet Organisationsaspekte für die Arbeitspakete, in die ein Projekt gewöhnlicherweise aufgeteilt ist, während die dritte Ebene die einzelnen Entwicklungsgruppen der Projektpartner betrachtet.

Projektebene: Die meisten Projekte verfügten über eine organisatorische Ebene des Managements und der Projektleitung. Dabei beschränkten sich die Aufgaben der Projektleitung häufig auf rein administratorische, vermittelnde und erst nachrangig leitende Tätigkeiten. Insgesamt gab die Projektleitung oft die grobe Zielrichtung der Entwicklungen im Projekt vor, achtete auf die Einhaltung von Zeitvorgaben und die Durchführung der Arbeiten im vereinbarten Rahmen. Zugleich gab es meist kein ausgeprägtes technisches Management, das den Software-Entwicklungsprozess kontrollierte. Bei dem Projekt TextGrid war dem gesamten Projekt ein internationaler Fachbeirat zugeordnet, der Entwicklungen überprüfte und unverbindlich bewertete.

Die im Projekt auftretenden Rollen wurden oft in Personalunion von Beteiligten des Projektes besetzt. So gab es folgende Rollen:

- Projektleitung
- Nutzer (zumeist beteiligt am Projekt, oft auch technisch involviert)
- Entwickler (überwiegend technische Partner)
- Tester (keine spezielle Trennung zwischen den Partnern)

Arbeitspaketebene: Die meisten Entwicklungen folgten einem dezentralen Entwicklungsansatz, der sich stark nach der Aufteilung der Aufgaben in einzelne Arbeitspakete richtete. Innerhalb der Arbeitspakete wurden einzelne Teilaspekte des Projektes von einigen Partnern bearbeitet. Hier gab es meist keine explizite Rollenverteilung. Innerhalb dieser Ebene gab es zusätzlich keine starke Organisationsstruktur. Die Partner waren gewöhnlicherweise gleichberechtigt beteiligt. Dies liegt in förderrechtlichen Gegebenheiten, die mit sich bringen, dass jeder Partner eigenständig auftritt und keine direkte Weisungsbefügnis zwischen den Projektpartnern besteht. Die offizielle Funktion der Arbeitspaketleitung war daher zumeist die Gesamtdarstellung in Richtung der Projektebene.

Partnerebene: Innerhalb des Projektes und auf der Ebene des Arbeitspaketes traten die Projektpartner meist durch Kleingruppen auf, die wiederum eine interne Organisationsstruktur hatten sowie ein eigenes Entwicklungsvorgehen verfolgten. Insbesondere das Softwaremanagement wurde auf dieser Ebene individuell durchgeführt, folgte jedoch oft agilen Ansätzen. Auch auf dieser Ebene gab es keine feste Rollenzuteilung. Diese flache Hierarchie führte zu kurzen Eskalationswegen.

1.2.2 Kommunikationsstruktur:

Im Zusammenspiel mit der Organisationsstruktur jedes Projektes spielt auch die interne Kommunikation eine große Rolle. In diesem Abschnitt sollen die von den Projekten umgesetzten Kommunikationswege aufgezählt werden.

Im Vordergrund der Koordinations- und Kommunikationsbemühungen standen regelmäßige Treffen der Partner. Diese Veranstaltungen fanden oft halb- oder vierteljährlich statt und dienten zur Präsentation von Ergebnissen und zur Festlegung nächster Entwicklungsschritte. Zusätzlich zu diesen Synchronisationsmaßnahmen wurden häufigere Treffen der folgenden Form durchgeführt:

- Monatliche Video- oder Telefonkonferenzen, um dringendere Probleme anzusprechen und feinere Planungen durchzuführen
- Häufiger Abstimmung zwischen technischen Partnern
- Direkte Kommunikation zwischen Partnern, die an einer gemeinsamen Entwicklung beteiligt waren.

1.2.3 Zusammenfassung und Meinungen

Die erfolgreichen Umsetzungen der akademischen Projekte haben gezeigt, dass eine Modularisierung von Arbeitsaufgaben derart möglich ist, dass kleine Gruppen diese unabhängig voneinander bearbeiten können. Dies setzt jedoch einerseits große Anstrengungen bei dem Entwurf einer geeigneten Architektur, andererseits eine gute Spezifikation von Schnittstellen voraus. Trotzdem ist ein Synchronisationsbedarf in den Kommentaren der Befragten festzustellen, der nicht von den bisherigen Organisationsstrukturen befriedigt werden kann. So werden folgende Punkte besonders häufig angesprochen:

- Intensive Interaktion zwischen Entwicklern wird als essentiell angesehen. Dies verhindert Mehrfachentwicklungen und Integrationsprobleme. Da diese Probleme insbesondere ab der Arbeitspaketebene aufwärts auftreten, wird auch hier das Konzept von Programmiersprints aus der agilen Programmierung (z.B. Scrum) als Technik vorgeschlagen.
- Gleichzeitig wird gefordert, die Entwicklung Team-orientiert und weniger Task-orientiert durchzuführen. In diesem Zusammenhang wird ein Kernteam der technisch hauptverantwortlich tätigen Partner vorgeschlagen, das eng auch über Einrichtungs- oder Institutsgrenzen hinweg zusammenarbeitet und die meiste Projektarbeitszeit in die Entwicklung investiert. Dies soll eine stark koordinierte und regelmäßige Arbeit sicherstellen.

Insgesamt besteht Bedarf an übergreifenden Prozessen. Diese sollen nach den Anforderungen der Befragten jedoch überwiegend agil sein und gleichzeitig eine schnelle Folge Ergebnissen sowie deren kontinuierliche Bewertung durch Nutzer ermöglichen (Rapid Prototyping).

1.3 Umgang mit Risiken

Risikomanagement wird in der einschlägigen Literatur zum Softwaremanagement als zentraler Punkt bezeichnet. In den untersuchten Gridprojekten wurden Aspekte eines solchen Managements identifiziert und umgesetzt. Dabei wurde jedoch auch klar, dass im Rahmen der Projektentwicklungen und der Förderstruktur eine von kommerziellen Projekten abweichende Herangehensweise gewählt wird.

1.3.1 Entdeckung von Risiken und Strategien zur Vermeidung:

Im Rahmen der untersuchten Gridprojekte sind verschiedene Aspekte bezüglich der Risikoentdeckung und -vermeidung zu identifizieren:

Evaluationsphase bei der Antragsstellung: Bereits bei der Antragstellung für ein geplantes Gridprojekt wurden in der Regel zentrale Evaluationen getroffen, um das Risiko im Projektverlauf zu minimieren. Neben ersten Technologiefestlegungen findet in dieser Phase insbesondere eine grobe Planung des Projektverlaufes und eine Planung der benötigten Mittel statt. Deshalb ist bereits zu diesem Zeitpunkt eine Risikoabschätzung für jeden einzelnen Projektpartner notwendig.

Evaluation bei Projektbeginn: In den meisten Projekten war zu Beginn der Bearbeitungsphase eine Evaluationsperiode vorgesehen, in der folgende Aspekte genau betrachtet wurden:

- Technologiefestlegung
- Abschätzung des Migrationsaufwandes
- Zeitliche Planung, Entwicklungsstufen (kontinuierlich auch während der Laufzeit)
- Prototyping zur besseren Aufwandsabschätzung

Auf der Grundlage dieser Untersuchungen wurden abschließende Entscheidungen für den weiteren Projektverlauf getroffen, aber auch sog. Fallbacklösungen konzipiert, die das Scheitern eines Projektzieles verhindern sollten.

Kooperation und Synergien: Im Rahmen des D-Grid-Verbundens haben einige Projekte auf die Ausnutzung von Entwicklungen und Erfahrungen anderer Projekte gesetzt, um das Risiko einer Fehlevaluierung einerseits und einer Doppelentwicklung andererseits zu verhindern. So wurde ebenfalls versucht auf existierende und ggf. gemeinsam entwickelte oder gepflegte Technologien zu setzen, insbesondere im Bereich der Grid-Middleware.

Gremien und Kommunikation: Einige Community-Grids haben die enge Kooperation von Technikern untereinander, sowie die Zusammenarbeit von Technikern und Nutzern als Weg zur Eindämmung von Risiken identifiziert. Die Bildung entsprechender Gremien und die verstärkte Zusammenarbeit im Projektverbund ermöglichte die frühe Erkennung, Diskussion und Behebung von möglichen Risiken in der Entwicklung.

1.3.2 Zusammenfassung und Meinungen

Die Risikoabschätzung im Vorfeld der Antragstellung wurde von vielen Community-Grids als ausreichendes Mittel zur ersten Risikoabschätzung und zur Risikobegrenzung angesehen. Insbesondere die Schätzung des Aufwandes und Festlegung der Projektkosten erlaubten eine weitgehende Eindämmung des Risikos. Der feste Förderrahmen erlaubte zumindest keine Überschreitung der veranschlagten Projektkosten.

Die zeitliche Planung der Entwicklungsphasen wurde in einigen Projekten als zu grob eingeschätzt. Diese Einteilung konnte weithin zu Verzögerungen im Projektablauf führen, da die Reaktionszeiten auf Fehlentwicklungen innerhalb der Entwicklungsphasen zu groß war. Hier ist nach Ansicht vieler Projekte die flexible Anpassung und Abstimmung von Projektzielen in kleineren zeitlichen Abständen notwendig.

Schließlich wird Kommunikation als integraler Aspekt der Risikovermeidung gesehen, einerseits zur Aufdeckung von Risiken, andererseits zur schnellen Reaktion auf und Behebung vorhandener Risiken im Projektverlauf.

1.4 Testumgebungen und Qualitätsmanagement

Testumgebungen, -verfahren oder ein dediziertes Qualitätsmanagement ist in den meisten Gridprojekten nicht in ausgeprägter Weise vorhanden. Trotzdem wurden einige Testverfahren eingesetzt und Umgebungen geschaffen, die unter diesem Aspekt betrachtet werden sollen.

Test- und Qualitätsziele:

Als Test- und Qualitätsziele wurden in den untersuchten Projekten fast ausschließlich zuvor identifizierte Testszenarien genutzt, um einerseits den spezifizierten Funktionsumfang zu überprüfen, andererseits funktionale und nicht-funktionale Anforderungen zu testen.

1.4.1 Testverfahren und Testumgebungen:

Im Folgenden sind einige der angewendeten Methoden zur Überprüfung von Funktionalität und Anforderungen aufgeführt:

- Übergreifende Integrationstests
- Anwendung der Use Cases
- Nutzungstests (Anwenderworkshops)
- Belastungstest
- Testumgebung bestehend aus Test- und Produktivumgebung (kein fester Migrationsprozess)
- Innerhalb der Entwicklergruppen:
  - Funktionale Test

  - Unit-Testing (JUnit, etc.)
  - Integrationstests mit Partnern
  - Tests zur Langzeitevaluierung (Testumgebung)
  - Randomized Testing
- Qualitätsmanagement auch durch Versionierung
- Proprietäre Build-Umgebung

1.4.2 Test- und Zertifizierungsprozesse in den Communities:

In den Projekten sind weitestgehend keine Prozesse zum Qualitätsmanagement zu finden. HEP-CG folgte einem fest von den LHC-Experimenten und dem WLCG vorgegebenen Prozess, der im Rahmen des TeraScale-Verbundes eine gewisse Zertifizierung erlaubte. MediGrid und TextGrid folgten einem Prozess zur Anwendungintegration in halbautomatischer bzw. automatischer Weise für die Aufnahme neuer Features in den Produktionsbetrieb.

1.4.3 Zusammenfassung und Meinungen

Keines der Projekte verfügt über ein ausgeprägtes Qualitäts- oder Testmanagement. Insbesondere sind keine übergreifenden Zertifizerungsprozesse zu identifizieren. Insgesamt sprechen sich viele Community-Grids für die Definition eines Projekt-internen Zertifizierungsprozesses aus. Dabei sollen auch Testmethoden für aufwändige Tests in hochverteilten Umgebungen einbezogen werden. Außerdem soll das Testvorgehen abhängig von der Projektphase anpassbar sein.

1.5 Lebenszyklus und Nachhaltigkeit

Der Aspekt des Software-Lebenszyklus wurde von keinem Projekt in ausreichender Weise adressiert. Für vorhandene Software gibt es bisher keine einheitliche Vorgehensweise zur Integration und Außerbetriebnahme. Einige Aspekte, die in den Projekten befolgt wurden, werden im Folgenden aufgelistet.

1.5.1 Deploymentzyklen:

- Bestimmt durch zeitliche Vorgaben (Meilensteine, Deliverables, Workshops und Präsentationen auf Konferenzen)
- Zeitspanne für Deployment durch Projektleitung festgelegt
  - insbesondere aber keine Koordination und Regelmäßigkeit
  - Koordination multilateral unter den Partnern
- Releasezyklen alle 8 Wochen (TextGrid), Beta-Version für vorherige Begutachtung
- Automatisiertes Deployment (MediGRID)

1.5.2 Betrieb und Wartung:

- Vereinbarung zur Aufrechterhaltung der Infrastruktur über Projektende hinaus (C3Grid, AstroGrid-D)
- kein fester Wartungsplan, benötigte Arbeiten werden durchgeführt
- Kommunikation und Statusmeldungen über Mailinglisten

1.5.3 End-of-Life:

- keine einheitliche Vorgehensweise, Partner/AP-spezifisch (C3Grid)
- kein zertifizierter Prozess (MediGrid)
- an Projektförderung gekoppelt

1.5.4 Meinung/Fazit/Verbesserung:

- Prozess zur Betriebssicherstellung und zur Wartung notwendig
- Prozess für Deployment und Roll-Out notwendig
- Sicherstellung von Betrieb und Wartung auch durch Kooperation mit anderen Projekten sichern

Kapitel 2

Spezifische Rahmenbedingungen in akademischen Community-Grids

Die im vorhergehenden Kapitel durchgeführte Darstellung der Softwareentwicklung in verschiedenen Community-Grids des D-Grid-Verbundes ermöglicht bereits an dieser Stelle die Feststellung einiger Rahmenbedingungen und Gegebenheiten, die für die Arbeitsorganisation und die zeitliche Strukturierung der Arbeit bestimmend sind. Das vorliegende Kapitel fasst die wichtigsten Feststellungen zusammen und stellt damit ein Rahmenwerk zur Verfügung, in das sich der später beschriebene Softwaremanagementprozess konzeptionell, organisatorisch und technisch einordnet. Dabei sei darauf hingewiesen, dass sich viele der ermittelten Aspekte aufgrund dreier zentraler Einflüsse ergeben:

1. Der Einfluss des Projektzieles, nämlich der Aufbau eines Grids als hochverteilte und damit stark modularisierte Infrastruktur, ermöglicht die Aufteilung in verschiedene organisatorische Einheiten und die Entkopplung der Entwicklung sowie die Wahl ähnlicher Konzepte und Vorgehensmodelle bei der Entwicklung.

2. Der Einfluss formaler Rahmenbedingungen, die insbesondere durch die Antrags- und Förderformalitäten des Geldgebers entstehen, forciert weitgehend ähnliche Organisationsstrukturen und vergleichbare Ansätze von Vorgehensmodellen.

3. Die weitgehend eigenständige Arbeit von Institutionen, Universitäten und Forschungseinrichtungen im Projektverbund resultiert ebenfalls in den meisten Community-Grids in ähnlichen Organisations- und Planungsstrukturen, Kommunikationswegen und Regeln der Kooperation untereinander.

Die sich aus den drei Aspekten ergebenden, weit verbreiteten Vorgehensweisen sollen im Folgenden aufgeführt und näher beleuchtet werden. Es wird hier noch einmal betont, dass einige der festgestellten Aspekte nicht auf das ebenfalls untersuchte HEP-CG zutreffen, da dies bereits einem speziellen übergeordneten Prozess folgt, der von dem umschließenden internationalen Forschungsprojekt vorgegeben wird.

2.1 Aspekte der Arbeitsorganisation

Die Besonderheiten der Arbeitsorganisation von akademischen Grid-Projekten sollen hier an Aspekten untersucht werden, die sich für die meisten Community-Grids in ähnlicher Weise darstellten. Dabei werden insbesondere die Aspekte der übergreifenden und internen Organisation des Softwaremanagements[1] sowie die Umsetzung von Kommunikationsstrukturen beleuchtet.

2.1.1 Projektstruktur und übergreifende Organisation des Softwaremanagements

In jedem untersuchten Projekt lassen sich ähnliche Aspekte der Arbeitsorganisation feststellen. Neben einer weitgehend flachen Hierarchie fällt eine starke Entkopplung verschiedener Entwicklungsbereiche auf, die sich in ihrer Modularisierung analog zur Architektur der zu entwickelnden Grid-Infrastruktur verhält. Da im allgemeinen eine Service-orientierte Architektur zur Umsetzung verschiedener Grid-Mehrwertdienste gewählt wird, können die Arbeiten an den unterschiedlichen Diensten weitgehend entkoppelt werden. Aus diesem Grund bilden sich oft Entwicklergruppen, die ein weitgehend abgeschlossenes Thema bearbeiten und so zum Gesamtprojekt beitragen.

Als einzige übergeordnete Hierarchieebene ergibt sich über den Entwicklergruppen eine Projektleitung, die entweder die technische Administration ebenfalls übernimmt oder diese Aufgaben an ein technisches Projektmanagement delegiert.

Aufgrund der formalen Richtlinien des Förderers ergeben sich jedoch einige Einschränkungen in der hierarchischen Ordnung des Projektes und der Bildung der Entwicklergruppen. Da formal alle Partner gleichberechtigt an den Projekten teilnehmen und neben dem übergreifenden Projektziel eigene wissenschaftliche und technische Ziele verfolgen, ist ein mit kommerziellen Unternehmen zu vergleichender Entwicklungsprozess nicht direkt umzusetzen. Einerseits verpflichten sich die beteiligten Partner zur Kooperation, gleichzeitig ist jedoch die Sanktionsmöglichkeit des Projektmanagements gegenüber einem einzelnen Partner eingeschränkt. Natürlich stellt die Eigenständigkeit eines jeden Projektpartners sicher, dass alle wissenschaftlich relevanten Ziele ausreichend berücksichtigt werden und nicht einzelne Aspekte der Innovation anderen untergeordnet werden. Trotzdem erschwert dieser Umstand die technische Projektdurchführung, stellt damit ein wichtiges Eigenstellungsmerkmal des Entwicklungsprozesses für akademische Grid-Projekte dar und ist somit ein wichtiger Punkt für die Integration in einen angepassten Prozess für das Software-Management. Es stellt den Prozess vor die Herausforderung, Richtlinien für das Entwicklungsvorgehen im Projekt zu finden, gleichzeitig aber die formalen Rahmenbedingungen einzubeziehen.

Eine weitere Besonderheit des Entwicklungsprozesses ist die Integration von Nutzern in das Projekt. Da die Initiative für jedes hier untersuchte Community-Grid aus dem Bedarf einer Gruppe von Wissenschaftlern entstanden ist, sind diese stark am Projekt beteiligt und in den Entwicklungsprozess integriert. Damit ergeben sich im Vergleich zu kommerzieller, Auftrags-basierter Softwareentwicklung Rahmenbedingungen, die grundsätzlich vorteilhaft für die direkte Integration von Nutzeranforderungen und die ständige Überprüfung der Entwicklungsziele sind. Jedoch nehmen die Nutzer oft auch die Rolle von Entwicklern ein. Für eine solche Konstellation gibt es in der bisherigen Praxis des Software Engineering fast überhaupt keine geeigneten Vorgehensmodelle.

[1] Eine umfängliche Darstellung der Organisationsstrukturen der Community-Grids ist in der Blaupause zu Organisationsstrukturen zu finden.

2.1.2 Kommunikationsstrukturen im Projektverbund

Die Etablierung geeigneter Kommunikationsstrukturen trägt auch in Grid-bezogenen Softwareentwicklungen ganz zentral zur erfolgreichen Umsetzung bei. Dies gilt insbesondere, da sich die Entwicklung, wie im vorherigen Abschnitt dargestellt, in viele Entwicklergruppen aufteilt. Die Herausbildung einer ansatzweise hierarchischen Organisationsstruktur ist dabei gewöhnlicherweise der Größe des Vorhabens geschuldet und stimmt mit der allgemeinen Theorie des Software Engineerings überein [2]. Dennoch zeigen sich auch hier Besonderheiten, die den eben genannten Prinzipien widersprechen und sicherlich teilweise aus den formalen Rahmenbedingungen der Projektförderstrukturen resultieren.

Aufgrund der weitgehenden Autonomie der beteiligten Projektpartner ist eine strikt hierarchische Organisation und damit eine Weisungsbefähigung des Projektmanagements nicht möglich. Entscheidungen müssen oft in einer eher demokratischen Weise zwischen allen beteiligten Partnern, unabhängig von den Rollen der Projektleitung, Nutzer und Entwickler, in einem Konsens gefunden werden. Aus diesem Grund haben viele Community-Grids spezielle Gremien eingerichtet, die die Koordinations- und Kommunikationsaufgabe zwischen den Projektpartnern übernehmen.

Globale Gremien sind für alle Projekte ein zentraler Weg zur demokratischen Abstimmung der Projektziele und zur Koordination der gemeinsamen Arbeit. Obwohl diese Gremien unterschiedliche Ausprägungen aufweisen (Steuerungsausschuss, Plenarmeetings, Videokonferenzen usw.), dienen sie alle als Kommunikationsplattform für die beteiligten, unabhängigen Projektpartnerpartner und als eigentliche Entscheidungsgremien. Damit stehen die globalen Gremien mindestens auf gleicher Hierarchieebene wie die Projektleitung und übernehmen so wichtige Aufgaben dieser Rolle. Sie können auch im Rahmen der Projekt-weiten Kommunikation gewisse Sanktionsmittel erhalten und einsetzen, die der Projektleitung, wie oben dargestellt, nicht immer zur Verfügung stehen. Ebenfalls sind in diesen Gremien die Partner mit Nutzerrolle vertreten und können so direkt Einfluss auf die Ziele des Projektes während der Ausführung nehmen. Aufgrund dieser Aspekte ist es wichtig, die besondere Funktion der globalen Gremien in einen Softwareprozess für das Grid zu integrieren.

Entwicklergremien bilden in den meisten Community-Grids die Kommunikationsebene zwischen den Projektpartnern, die die Rolle technischer Entwickler übernehmen. Diese Gremien sind insbesondere für die Synchronisation von Entwicklungsdetails unter den oft entkoppelt arbeitenden Entwicklergruppen notwendig. Zugleich bieten die Entwicklergruppen eine Plattform für die Nutzer, direkten Einblick in technische Entwicklungen und zugleich Einfluss darauf zu nehmen. Auch die Entwicklergremien entscheiden rein demokratisch, da alle Partner gleichberechtigt sind.

2.2 Zeitliche Aspekte

Die zeitlichen Rahmenbedingungen eines Community-Grid-Projektes werden ebenfalls stark durch formale Rahmenbedingungen der Fi¿½rderer bestimmt. Neben der initialen Antragsphase, die den konzeptionellen Start der Projektarbeit beschreibt, sind feste Zeitplanungen im Antrag und regelmi¿½i¿½ige Lieferungen von Ergebnissen bereits durch Fi¿½rderrichtlinien vorgeschrieben. Diese lassen sich nur in begrenzter Weise veri¿½ndern und geben somit eine zeitliche Grobstruktur vor. Auch der Einfluss von Nutzern auf die zeitliche Planung ist aufgrund der direkten Integration in das

Projekt hervorzuheben. Im Folgenden werden diese Besonderheiten der Softwareentwicklung fi¿½r Grid-Infrastrukturen genauer betrachtet.

2.2.1 Rahmenwerk des Projektantrags

Der Projektantrag bietet neben den inhaltlichen Richtlinien und den zumeist funktionalen Anforderungen[2] vor allem eine zeitliche Planungsvorgabe des Projektverlaufes und einen Startpunkt fi¿½r die Projektarbeit.

Da der Projektantrag eine formale Voraussetzung fi¿½r die Fi¿½rderung eines Projektes ist und gleichzeitig als Entscheidungsgrundlage fi¿½r die finanzielle Unterstï¿½tzung des Vorhabens dient, ist bereits zu Beginn des Projektes ein ausfi¿½hrlicher Arbeitsplan vorzulegen. Sind auch die technischen Ziele des Projektes zu diesem Zeitpunkt nicht endgï¿½ltig festgelegt, so ist dennoch eine Angabe von Lieferterminen (Deliverables) und Meilensteinen (Milestones) vorgesehen. Diese definieren angestrebte, aber noch nicht vollstï¿½ndig spezifizierte Zwischenziele. Die Lieferungen kï¿½nnen gewï¿½hnlicherweise wï¿½hrend des Projektverlaufes nur schwer, und dann nur gut begrï¿½ndet, verï¿½ndert werden. Daher mï¿½ssen diese festgelegten Zeitpunkte im Software-Managementprozess berï¿½cksichtigt werden.

2.2.2 Zeitplanung und Iteration im Projektverlauf

Die Umsetzung einzelner Projektziele wurde in den meisten Projekten nach festgelegten Meilensteinen ausgerichtet. Fï¿½r einen aktuell zu bearbeitenden Meilenstein wurde die Zeitplanung konkret ausgearbeitet, indem eine Projekt-ï¿½bergreifende Planungsphase eingefi¿½hrt wurde. Die Einhaltung des Projektplans wurde vom Projektmanagement ï¿½berwacht und der Projektfortschritt iterativ in globalen Gremien ï¿½berprï¿½ft. Die iterative ï¿½berprï¿½fung wird ebenfalls durch ein regelmï¿½ï¿½iges Berichtswesen unterstï¿½tzt. So sind nach Fï¿½rderrichtlinien regelmï¿½ï¿½ig (monatlich) Statusberichte an den D-Grid Beirat ï¿½ber den Projektfortschritt an den Projekttrï¿½ger zu ï¿½bermitteln. Ebenso ist ein halbjï¿½hrlicher und ausfi¿½hrlicher Zwischenbericht an den Projekttrï¿½ger zu richten (durch Projektpartner und als zusï¿½tzlicher Statusbericht durch den Projektleiter). Dies bietet grundsï¿½tzlich ein straffes Rahmenwerk fi¿½r die Durchfi¿½hrung des Projektplans und kann vom Software-Managementprozess integriert werden.

2.2.3 Einfluss der Nutzer

Als Besonderheit der Organisation ist bereits die Integration der Nutzer in die konzeptionelle und technische Arbeit eines Grid-Projektes herausgestellt worden. Dies hat natï¿½rlich auch direkte Auswirkungen auf die Zeitplanung. Insbesondere die Mï¿½glichkeit Nutzerfeedback direkt in den Entwicklungsprozess einflieï¿½en zu lassen, ermï¿½glicht eine schnelle Anpassung an verï¿½nderte oder verfeinerte Anforderungen. Dies setzt aber auch eine gewisse Flexibilitï¿½t des Entwicklungsprozesses voraus, bzw. macht eine klare Regelung zum Umgang mit einflieï¿½endem Feedback zu einem wichtigen Bestandteil des Software-Managements. Eine unkontrollierte und wenig geregelte ï¿½bernahme von Anforderungsï¿½nderungen und Feedback in den laufenden Entwicklungsprozess

[2] Funktionale Anforderungen beschreiben, welche Funktionalitï¿½t das zu erstellende System besitzen soll. Sie kï¿½nnen durch eine Spezifikation von erwarteten Ein- und Ausgabebedingungen festgelegt werden und werden im Entwurf durch UseCase-Diagramme dargestellt.

kï¿½nnte zu starken zeitlichen Verzï¿½gerungen im Entwicklungsvorgehen fiï¿½hren und damit die strengen Rahmenbedingungen des vorhergehenden Abschnittes verletzen.

Kapitel 3

Ein Software-Managementprozess für das Grid

Ein Software-Prozess ist definiert durch die formale Beschreibung eines Ablaufes für die Durchführung eines Vorhabens [2]. Dabei stehen vier Aspekte im Vordergrund: die Beschreibung der durchzuführenden Schritte, die an der Durchführung beteiligten Personen, die für den Ablauf benötigten und von ihm erzeugten Informationen sowie das zeitliche Zusammenspiel der zuvor genannten Elemente.

Dieses Kapitel entwickelt einen Prozess, der für die Softwareerstellung im Grid jeden der genannten Punkte adressiert und ihnen einen formalen Rahmen gibt. Dabei werden Elemente bekannter Software-Prozesse aufgegriffen und in das Konzept integriert. Allgemein kann der vorgeschlagene Prozess als iterativer und inkrementeller Ansatz gesehen werden, der in vielen Aspekten den von Lehman [5] beschriebenen Ansatz aufgreift. Insbesondere ist dieser Prozess aber auf Besonderheiten der Softwareentwicklung in Grids zugeschnitten und kann an vielen Stellen flexibel konfiguriert und instanziiert werden.

3.1 Ein Meilenstein-basierter Softwareevolutionsprozess für das Grid

Ausgehend von den Rahmenbedingungen der Softwareentwicklung im Grid und den Erfahrungen der verschiedenen Community-Grids bezüglich des Entwicklungsprozesses, wird ein iterativer, evolutionärer Prozess vorgeschlagen. Dieser ist angelehnt an die Arbeiten von Lehman [5, 6, 7] und wird von Glinz [2] auch allgemein als Wachstumsprozess bezeichnet.

Die Grundidee dieses Prozesses, siehe Abbildung 3.1, entspricht dem evolutionären Wachstumsgedanken: eine Software wird nicht vollständig und im Vornherein konstruiert, sondern wächst während des Entwicklungsprozesses. Ausgangspunkt für einen solchen Prozess sind die Kernziele eines Vorhabens, die in einer groben Aufstellung der umzusetzenden Funktionalität und Ziele münden, sowie eine Folge von Lieferungen und Lieferterminen definieren (Meilensteine). Dabei sind die Lieferungen nicht von Beginn an über den gesamten Projektverlauf vollständig spezifiziert, sondern werden während der Umsetzung des Projektes schrittweise konkretisiert, fertiggestellt und veröffentlicht. Trotzdem besteht immer die Anforderung, dass eine Lieferung vollständig funktionsfähig (also nutzbar) sein soll. Sie wird dann als aktuelle Generation des Produktes eingeführt. Bei der Entwicklung

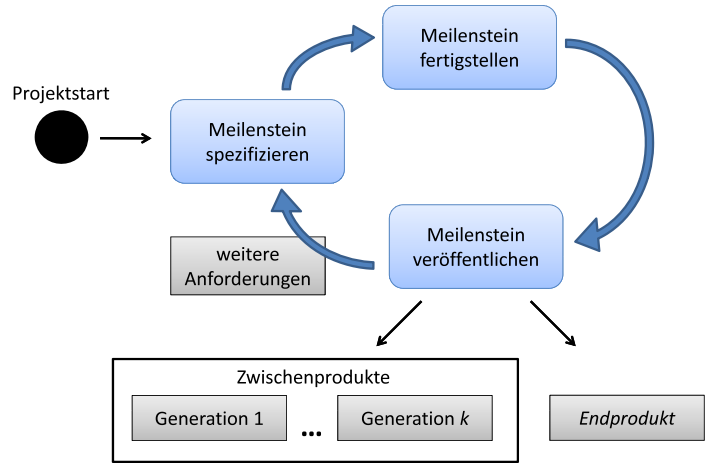

Abbildung 3.1: Schematische Darstellung des übergreifenden Prozessablaufes. Die Evolutionsschleife besteht aus drei Meilenstein-getriebenen Schritten, die die Erstellung einer neuen Softwaregeneration beschreiben.

gemachte Erfahrungen und sich aus der Nutzung der aktuellen Generation ergebende zusätzliche Anforderungen können dann in einem kontinuierlichen Integrationsprozess leicht in den weiteren Projektfortschritt eingebunden werden.

Formaler ausgedrückt ist der Entwicklungsprozess in eine Folge von Iterationen unterteilt, wobei nach jeder Iteration ein funktionsfähiges und vollständiges Teilergebnis der zu entwickelnden Software vorliegt und ausgeliefert wird. Jede Teillieferung ist als ein weitgehend eigenständiges Teilprojekt organisiert und kann verschiedene Softwareprozesse (Phasen-basierte Modelle, agile Entwicklung) umsetzen. Diese so genannten *internen Prozesse* werden in dem übergeordneten evolutionären Prozess nicht vorgegeben und können individuell umgesetzt werden.

Das Modell scheint deshalb gut auf Grid-Softwareentwicklungen anwendbar, da es weitgehend das natürliche Entwicklungsvorgehen bei großen Systemen abbildet. Zusätzlich erlaubt es die flexible Einteilung von Projektbestandteilen in einzelne Teilprojekte und damit einen einfachen Weg zur Projektsteuerung. Gleichzeitig steht schon sehr früh ein funktionsfähiges System (Pilotanwendung) zur Verfügung, das der starken Einbindung von Nutzern im Projekt und der Integration von Nutzerfeedback entgegenkommt.

Auf der anderen Seite setzt die Entwicklung ein sehr systematisches Vorgehen und eine sorgfältige Planung voraus, damit das Projekt nicht einerseits in viele Teilprojekte zerfällt und andererseits die Lieferungen nicht in Zielsetzung und Umsetzung derart divergieren, daß eine vernünftige Integration und Pflegbarkeit des Gesamtsystems unmöglich wird. Zusätzlich setzt das Vorgehen eine relativ umfassende, wenn auch nicht sehr präzise Definition der zu erreichenden Ziele voraus, um während der evolutionären Schritte nicht durch starke Abweichungen den Evolutionspfad zu verlassen und ein Redesign des Gesamtsystems vornehmen zu müssen.

Um den aufgeführten Risiken zu begegnen, stellt das vorliegende Kapitel eine ausführliche und auf Entwicklungen im Grid-Kontext angepasste Definition des Softwareprozesses zur Verfügung. Dabei werden zuerst wichtige Grundbegriffe definiert und anschließend Rollen, Informationen, Aktivitäten und Abläufe angegeben.

3.1.1 Definitionen

In diesem Abschnitt werden die im Folgenden verwendeten Grundbegriffe definiert und die dazugehörige grafische Notation erläutert. Diese Elemente sind für die spätere Beschreibung der Aktivitäten und die Darstellung der Abläufe wichtig.

Grundbegriffe

Einleitend sind vier Aspekte eines Softwareprozesses angegeben worden, die sich auf die verantwortlich Handelnden, durchzuführende Tätigkeiten, Informationen und zeitliche Abläufe beziehen. Diese lassen sich formal in folgende Konstrukte fassen:

Rollen bezeichnen Verantwortlichkeiten von im Prozess Handelnden. Dabei ist eine Rolle nicht auf eine einzige Person beschränkt. Auch Gruppen können die Funktion einer Rolle übernehmen und die damit verbundenen Verantwortlichkeiten und Aufgaben teilen. Ebenso ist es möglich, dass mehrere Rollen von einer einzigen Person oder der gleichen Gruppe eingenommen werden. Die genaue Zuordnung der Rollen im Projektteam ist nicht Bestandteil des Prozesses. Obwohl die Verantwortlichkeiten und Tätigkeiten einer Rolle im Prozess definiert werden, ist die personelle Zuordnung zu Beginn des Projektes oder einer Projektphase von der Projektleitung zu treffen. Im speziellen Fall der Grid-Projekte wird eine solche Zuordnung oft bereits im Rahmen der Antragsphase festgelegt.

Aktivitäten bezeichnen die von einer Rolle durchzuführenden Tätigkeiten. Deshalb wird einer Aktivität generell eine Rolle zugeordnet, die diese ausführt und dabei ein kleine Anzahl von Artefakten nutzt, verändert oder erzeugt. Jede Aktivität kann als abstrakte Handlung verstanden werden, die selbst wieder einen komplexen Aufbau hat, wobei es notwendig ist, sie auf einem bestimmten Betrachtungslevel des Prozesses als atomare Handlung zu verstehen, um den Prozess nicht zerfasern zu lassen. Die Literatur rät eine Aktivität in ihrer zeitlichen Komplexität nicht über wenige Tage hinaus und nicht unter wenigen Stunden zu definieren. Für eine einheitliche Darstellung nehmen wir hier jedoch verschiedene Komplexitätsebenen für eine Aktivität an. So tauchen in der folgenden Prozessbeschreibung für das Grid-Softwaremanagement Aktivitäten auf, die sich über einen langen Zeitraum erstrecken, jedoch im Sinne von Workflows in Unteraktivitäten unterteilt werden können.

Artefakte bezeichnen in einen Prozess einfließende und aus dem Prozess entstehende Informationen und Produkte. Sie stehen den Rollen zur Ausführung einer Aktivität zur Verfügung und werden ggf. von einer Aktivität produziert. Dabei kann es sich um einfache Informationen, komplexe Produkte, Dokumente oder auch das schließliche Endprodukt des Prozesses handeln. Generell werden Artefakte genutzt, modifiziert oder erzeugt. Dabei kann implizit eine Hierarchie oder Teil-von-Beziehung angenommen werden: Rollen erzeugen auf Basis existierender Artefakte neue Artefakte während ihrer Aktivitäten. Bei dem Endprodukt handelt es sich folglich um das umfassenste Artefakt, die Gesamtmenge aller Artefakte.

Workflows beschreiben das Zusammenwirken der zuvor definierten Elemente. Ohne einen Zusammenhang wäre der Pfad der Produkterstellung durch Rollen, Aktivitäten und Artefakte nicht ausreichend spezifiziert. Die Workflows geben den zeitlichen Zusammenhang zwischen Aktivitäten an, bestimmen Verantwortlichkeiten und Zusammenspiel von Rollen und beschreiben den Informationsfluss und Produktentstehung (Artefakte) im Prozess. Bei der Darstellung

von Zusammenhängen können jedoch unterschiedliche Detailgrade genutzt werden. Deshalb ist nicht immer jede Interaktion und jede Folge von Aktivitäten berücksichtigt, sondern Teil der Ausgestaltung des Prozesses, insbesondere der Rollenverteilung.

Grafische Notation

Zur Darstellung eines Prozesses wird gewöhnlicherweise auf eine grafische Darstellung des Workflows zurückgegriffen, da dieser sowohl inhaltliche Aspekte (Aktivitäten, Rollen und Artefakte), wie auch zeitliche Aspekte zusammenfasst. Abbildung 3.2 fasst die in diesem Dokument zur Verfügung stehenden grafischen Elemente zusammen.

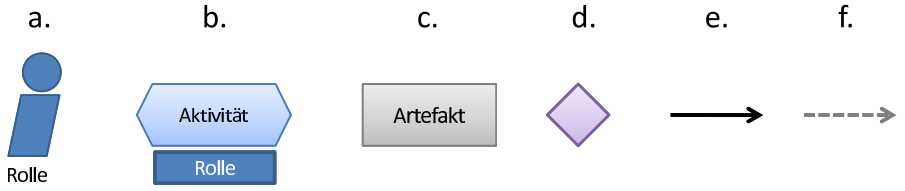

Abbildung 3.2: Legende der verwendeten grafischen Notation: (a) Rollen im Prozess, (b) Aktivitäten mit beteiligter Rolle, (c) Artefakte, Produkte, (d) Entscheidungspunkte, (e) zeitliche Abhängigkeiten, (f) Artefaktgenerierung.

Rollen werden grafisch als stilisierte Personen dargestellt (Abb. 3.2a), können jedoch der Übersichtlichkeit halber auch an ausgeführten Aktivitäten annotiert werden. Aktivitäten werden als atomare Handlung dargestellt, die von einer oder mehreren Rollen ausführbar sind (Abb. 3.2b). Artefakte werden als Rechtecke (Abb. 3.2c) dargestellt und knapp mit ihrem Informationsgehalt bezeichnet. Als zusätzliche grafische Elemente tauchen Diamanten auf, die zur Konstruktion von Bedingungen und des zeitlichen Ablaufes notwendig sind. Diese Entscheidungspunkte (Abb. 3.2d) erlauben das Einschließen von Entscheidungen und Verzweigungen in Workflows. Der zeitliche Ablauf wird durch gerichtete Kanten zwischen den Aktivitäten gekennzeichnet (Abb. 3.2e), während Informationseinfluss und die Erstellung von Informationen und Produkten über gestrichelte Kanten dargestellt wird (Abb. 3.2f).

3.1.2 Artefakte im Prozess

Die im Rahmen des Prozesses genutzten und entstehenden Artefakte werden in diesem Abschnitt näher beschrieben. Dabei liegt ein besonderer Fokus auf den Ergebnissen des Entwicklungsprozesses. Neben den Produktbausteinen und den Lieferobjekten, werden vor allem die Meilensteine als den Prozess antreibende Elemente betrachtet.

Produktbaustein: Dieses Artefakt bezeichnet im Softwareentwicklungsprozess einen atomaren, von einem Beteiligten erstellbaren Teil des Gesamtproduktes. Es ist in seiner Eigenschaft als Bestandteil des Produktes nicht mehr in Untereinheiten aufteilbar und wird von der verantwortlichen Rolle als Ergebnis abgeliefert. Der Umfang eines Produktbausteins muss also derart gewählt werden, dass seine Bearbeitung mit geringem Zeitaufwand bewältigt werden

kann. Andernfalls ist eine große Verzögerung des Prozessablaufes durch einen einzigen Produktbaustein nur schwer zu verhindern. Im Prozess wird weiterhin angenommen, dass jeder Produktbaustein einen Teil eines Lieferobjektes darstellt. Inhaltlich ist jedoch nicht spezifiziert, worin ein Produktbaustein besteht. So können diese Produktbausteine z.B. Untersuchungsergebnisse, Anforderungssammlungen, Spezifikationen oder auch Implementierungen sein. Die Verantwortung für einen Produktbaustein liegt im vorgeschlagenen Prozessentwurf zumeist bei der Rolle des Fördernehmers, der vom Förderer für seine Arbeit am Projekt finanzielle Mittel erhält. Aufgrund der feinen Granularität des Produktbausteins ist er normalerweise nicht Bestandteil des Projektantrages, sondern wird erst während des Prozesses genau spezifiziert.

Lieferobjekt: Dieses Artefakt stellt ein komplexeres Produkt dar, das gewöhnlicherweise aus verschiedenen Produktbausteinen zusammengesetzt ist. Es stellt ein abgeschlossenes Teilergebnis eines Meilensteins dar und taucht als solches üblicherweise bereits im Projektantrag auf. Die Erstellung eines Lieferobjektes basiert auf der koordinierten Fertigstellung von Produktbausteinen und erfordert deshalb eine besondere Aufmerksamkeit im Entwicklungsprozess, um Verzögerungen im Gesamtablauf zu verhindern. Darum wird die Bearbeitung eines solchen Lieferobjektes oft in Arbeitspakete oder Arbeitsgruppen unterteilt, wobei die Koordination der Arbeit von der Rolle des Arbeitspaketmanagements übernommen wird. Beispiele für ein Lieferobjekt sind einzelne abgeschlossene Softwarekomponenten, Architekturspezifikationen, Analysen und umfassende Berichte.

Meilenstein: Die formale Definition von Meilensteinen sieht diese als Stellen in jedem Prozess, an denen ein geplantes Ergebnis vorliegt. Darüber hinaus sind sie im vorliegenden Prozess jedoch die zentralen, den Ablauf strukturierenden und antreibenden Einheiten. Sie stellen das Zwischenergebnis der Entwicklung dar, umfassen Lieferobjekte und Produktbausteine und geben zugleich den Entwicklungshorizont der aktuellen Iteration vor. Die Fertigstellung und Veröffentlichung eines Meilensteins entspricht hier der Bereitstellung einer neuen Generation der zu entwickelnden Software. Damit kann der Meilenstein als ein Teilprojekt im Gesamtvorhaben gesehen werden. Jeder neue Meilenstein setzt also eine Planungs-, Fertigstellungs- und Überprüfungsphase voraus, die jeweils individuell und abhängig von Inhalt und Zielen umgesetzt werden kann. Für die Bearbeitung eines Meilensteins ist das Projektmanagement zuständig.

Bericht/Dokument: Diese Artefakte sind Informationen und Produkte des Entwicklungsprozesses, die nicht immer unmittelbar als Produktbausteine in die anderen Artefakte integriert werden und in den weiteren Entwicklungsprozess einfließen. Die formalen Rahmenbedingungen öffentlich geförderter Projekte erfordern regelmäßige Status- und Tätigkeitsberichte, um die Verwendung der Fördermittel nachvollziehbar zu machen. Daher ist die Erstellung zwar unmittelbar in den Prozess integriert, die Informationsverwendung kann aber im Softwareentwicklungsprozess nur selten einer beteiligten Rolle zugeordnet werden.

3.1.3 Allgemeine Rollen im Prozess

Während des Prozesses treten eine Vielzahl allgemeiner Rollen in Aktion, die unter diesem Abschnitt kurz genannt, beschrieben und bezüglich ihrer Verantwortlichkeiten erläutert werden. Zusätzlich werden Vorschläge für die Instanziierung der jeweiligen handelnden und beteiligten Einheiten gemacht. Dies sind jedoch Empfehlungen, deren Umsetzung im Rahmen der Prozesskonkretisierung speziell an die Anforderungen angepasst werden können.

Projektmanagement: Das Projektmanagement hat im gesamten Prozess eine doppelte Aufgabenstellung. Einerseits liegt bei ihm die formale Leitung der Projektes im Sinne der Prozesssteuerung und -überwachung. Andererseits sind mit der Rolle administrative Verantwortlichkeiten verbunden. Im Meilenstein-getriebenen Prozess zur Softwareentwicklung besteht die Aufgabe des Projektmanagements in der Einleitung verschiedener Arbeitsphasen und deren Überwachung. So ist die Rolle dafür verantwortlich, dass der in Abbildung 3.1 dargestellte Entwicklungszyklus vorangetrieben und eingehalten wird. Dazu muss jede der Meilenstein-bezogenen Aufgaben, die Spezifikation, Fertigstellung und Veröffentlichung von dem Projektmanagement angestoßen und betreut werden. Gleichzeitig ist die Rolle für die administrative Durchführung des Berichtwesens verantwortlich. Dies umfasst die Erfassung und Aggregation von technischen Entscheidungen, Projektfortschritten, Problemen im Projektverlauf und allgemeinen Berichten im Rahmen der Förderbestimmungen.

Die Besetzung der Rolle des Projektmanagements wird oft in unterschiedlicher Form realisiert. Neben einer einzelnen Institution, die im Projektantrag für die Managementaufgaben spezielle Mittel beantragt und das Projekt zugleich in der formalen Funktion des Konsortialleiters vertritt, kann auch eine Gruppe aus Vertretern aller beteiligter Institutionen die Verantwortlichkeit des Projektmanagements übernehmen. Wie bereits in Kapitel 2 diskutiert, ergeben sich aus den Besetzungsszenarion und den Rahmenbedingungen unterschiedliche Problemstellungen: Eine Besetzung der Rolle durch einen einzigen Partner schränkt die Steuerungsmöglichkeiten des Verbundes ein, der oftmals aus Partnern besteht, die Eigeninteressen verfolgen und formal eigenständig am Projekt teilnehmen. Die Etablierung eines steuernden Gremiums hingegen wird zu meist einvernehmlichen Entscheidungen führen, jedoch auch zu einem im demokratischen Entscheidungsfindungprozess begründeten gesteigerten Zeitaufwand führen. Eine genaue Betrachtung der Organisationsformen in Grid-Projekten ist in der zugehörigen Blaupause zu finden.

Arbeitspaketmanager: Die Verantwortlichkeiten des Arbeitspaketmanagers liegen auf der Ebene der Lieferobjekte, die jeweils als Bestandteile der den Projektverlauf bestimmenden Meilensteine betrachtet werden können. Diese Rolle umfasst die Spezifikation, Überwachung und Zusammenführung der Lieferobjekte und ihrer Produktbausteine. Gleichzeitig besitzt auch das Arbeitspaketmanagement eine geringe administrative Komponente, die sich wie beim Projektmanagement auf die Erstellung formaler Berichte bezieht, um die Arbeiten gegenüber dem Projektmanagement zu dokumentieren.

Die Rolle des Arbeitspaketmanagers wird im Kontext der Grid-Softwareentwicklung häufig durch den Repräsentanten eines Projektpartners besetzt und umfasst die Betreuung mehrerer thematisch ähnlicher Lieferobjekte, die in einem Arbeitspaket zusammengefasst werden. Eine Besetzung der Rolle durch ein Gremium ist ausschließlich bei einem umfangreichen Arbeitspaket mit Beteiligung vieler Fördernehmer mit einzelnen Interessen sinnvoll, da sonst ein starker Synchronisationsoverhead entsteht, der zu Verzögerungen auf der Arbeitsebene führen kann.

Fördernehmer: Die Rolle des Fördernehmers repräsentiert diejenigen Beteiligten am Projekt, die die Projektarbeit im Sinne der Erzeugung von Produktbausteinen übernehmen. Ein Fördernehmer stellt die atomaren Artefakte zur Verfügung, aus denen einzelne Lieferobjekte zusammengesetzt werden. Seine Verantwortlichkeit ist etwa mit der Rolle der Workers im Rational Unified Process [3] zu vergleichen und kann viele spezielle Ausprägungen annehmen, die teilweise im Folgenden diskutiert werden und teilweise in der Spezifizierung verschiedener Aktivitäten definiert werden müssen.

Diese Rolle wird gewöhnlicherweise von jedem im Projekt geförderten Partner eingenommen. Dabei ist natürlich nicht festgelegt, ob eine einzelne Person oder eine Entwicklergruppe die Verantwortung für die Aufgaben übernimmt. Jedoch sind die Tätigkeiten eines Fördernehmers im vorliegenden Prozess immer entwicklungsbezogen und tragen zur Fertigstellung eines Produktbausteines bei.

Nutzer: Die eigentliche Zielgruppe des Softwareentwicklungsprozesses wird unter der Rolle des Nutzers zusammengefasst. Der Nutzer ist nicht an der eigentlichen Entwicklungsarbeit beteiligt, liefert im Prozess jedoch an verschiedenen Stellen Feedback zu Zwischenergebnissen der Entwicklung.

Im Kontext von Grid-Projekten wird die Rolle von Nutzern neben externen Interessenten durch Projektpartner selbst besetzt, siehe Kapitel 2. Dabei können verschiedene Interessen in der Community durch verschiedene Partner vertreten werden, so dass die Nutzerrolle gewöhnlich von einem Gremium übernommen wird, das Anforderungen und Feedback in gesammelter Form in den Entwicklungsprozess einbringt.

3.1.4 Besondere Rollen im Prozess

Neben den oben vorgestellten allgemeinen Rollen des Softwareentwicklungsprozesses tauchen zusätzliche Rollen auf, die eine eher funktionale Charakteristik im vorgeschlagenen Vorgehen haben. Sie bilden an wichtigen Stellen der Systemkonzeption, Produktzusammenführung, -überprüfung und -veröffentlichung eine zentrale Rolle.

Systemanalytiker: Die Rolle des Systemanalytikers ist für die Durchführung der Anforderungsanalyse zuständig und modelliert unter Einbeziehung der Nutzer die Anwendungsfälle zur Beschreibung der Systemfunktionalität. Damit wird das Softwaresystem als Ganzes erfasst und von der Außenwelt, also nicht zum System gehörenden äußeren Elementen und Zusammenhängen, abgegrenzt. Im vorliegenden Prozess tritt der Systemanalytiker an zwei Stellen in Aktion. Er ist einerseits an der genauen Spezifikation eines Meilensteins beteiligt und andererseits während der Veröffentlichung des Meilensteins in eine eventuelle Korrekturphase eingebunden.

Aufgrund der Rahmenbedingungen der Grid-Projekte sollte die Rolle des Systemanalytikers durch mehrere Personen besetzt werden, die einerseits der Nutzerschaft, andererseits den technischen Partnern entstammen. Durch diese Kombination von Nutzer- und Entwicklereinfluss kann eine effiziente Abgrenzung des Systems erreicht werden, die so schneller in die Konzeptionsphase der Entwicklung übernommen wird.

Integrator: Die Verantwortung der Rolle des Integrators liegt in der Ergebniszusammenführung während des Entwicklungsprozesses. Sie ist für die Kompilation der Lieferobjekte zu einem Meilenstein zuständig und verwendet als Input die während einer Iteration erstellten Lieferobjekte. Im Entwicklungsprozess stellt die Tätigkeit der Rolle einen Synchronisierungspunkt für alle durchgeführten Arbeiten dar und steht deshalb einerseits in enger Beziehung mit dem Projektmanagement, andererseits ist sie auf die Lieferungen der Arbeitspaketmanager und Fördernehmer angewiesen.

In der Grid-Umsetzung des Prozesses ist diese Rolle selten durch eine einzelne Person oder Institution besetzt. Vielmehr vereint sie oft Vertreter aus dem Arbeitspaketmanagement und

aus Reihen der Fördernehmer zu einer Gruppe, die mit der Integration der Ergebnisse beauftragt werden. Insgesamt macht eine Synchronisation der an der Entwicklung der zusammenzuführenden Lieferobjekten Beteiligten Sinn, um schnelle Anpassungen für eine reibungslose Integrationsarbeit zu ermöglichen. Jedoch muss es einen Beauftragten oder ein Gremium für die Planung und Überwachung der Durchführung der Integration geben, ebenso wie einen spezifizierten Prozess.

Tester: Die Rolle des Testers ist mit der Überprüfung der Entwicklungsergebnisse verbunden. Sie stellt fest, ob ein Meilenstein *technisch* funktionsfähig ist. Diese Überprüfung umfasst insbesondere die Planung und Durchführung der technischen Abnahme des Gesamtsystems, kann sich aber auch auf die das Produkt bildenden Lieferobjekte ausdehnen. Die Aufgabe des Testers ist jedoch nicht, alle Produktbausteine zu überprüfen. Dies liegt im Zuständigkeitsbereich der Fördernehmer und des Arbeitspaketmanagers.

In vielen Fällen sollte auch die Rolle des Testers in Grid-Projekten durch ein Gremium aus der Gruppe der Fördernehmer besetzt werden, um einen direkten Rückfluss an Informationen zur Fehlerbehebung zu ermöglichen. Somit ist die Tätigkeit zeitlich nicht immer von der des Integrators zu trennen. Hier sind also Schnittpunkte zwischen den beiden Rollen des Integrators und des Testers zu finden, die bei der konkreten Umsetzung des Prozesses berücksichtigt werden müssen.

Releasemanager: Die Veröffentlichung des Produktergebnisses wird von der Rolle des Releasemanagers übernommen. Diese Rolle organisiert die Überführung des fertigen Zwischen- oder Endproduktes in ein Produktivsystem und schließt damit die Erstellung des Meilensteins ab. Dazu können eigene Prozesse definiert werden, die eine zertifizierte Veröffentlichung der neuen Softwaregeneration sicherstellen.

An der Ausfüllung dieser Rolle sollte im Grid-Kontext eine Gruppe von Projektpartnern beteiligt sein, die unter Aufsicht des Projektmanagements entsprechende Releasepläne entwickelt und durchführt.

3.1.5 Grundlegende Workflows

Die zuvor aufgeführten und beschriebenen Prozesselemente stellen wichtige Bestandteile des gesamten Softwareentwicklungsprozesses dar, sind jedoch bisher in keinen direkten Zusammenhang gebracht worden. Da eine Erfüllung und Nutzung der obigen Strukturen nur im zeitlichen Projektverlauf sinnvoll ist, schließt dieser Abschnitt die Definition des grundlegenden Softwareprozesses mit der Angabe zeitlicher Zusammenhänge in Form von Workflows ab. Die hier vorgestellten drei Workflows spezifizieren auf höchster Ebene den Entwicklungsfortgang im evolutionären Prozess aus Abbildung 3.1. Sie legen explizit die Verantwortlichkeiten der Rollen, die zeitliche Reihenfolge der Tätigkeiten und die Erzeugung und Verwertung von Ergebnissen (Artefakten) fest.

Der übergeordnete evolutionäre oder iterative Entwicklungsprozess gibt eine aufeinander aufbauende Erstellung von funktionsfähigen Zwischenprodukten vor, die jeweils durch einen Meilenstein definiert werden. Der Prozess ist also in jedem Generationsschritt ausschließlich auf die Fertigstellung des aktuellen Meilensteins konzentriert. Zu Beginn einer jeden Iterationsphase muss der im Projektplan grob vorgegebene Meilenstein detailliert spezifiziert werden. Danach ist eine Phase der Meilenstein-Bearbeitung und -Fertigstellung vorgesehen. Die Iteration wird dann von einer letzten Phase der Meilenstein-Veröffentlichung abgeschlossen. Im Folgenden werden die drei Phasen ausführlich in einem Diagramm dargestellt und kurz textuell beschrieben.

Meilenstein spezifizieren

Die Spezifikation eines Meilensteins leitet die Erzeugung einer neuen Softwaregeneration im vorgeschlagenen Prozess ein. In dieser Phase werden die für den Bearbeitungszeitraum im Projektantrag vorgesehenen Ziele und damit verbundenen Tätigkeiten genauer spezifiziert und zu einer *Spezifikation des Meilensteins* als Artefakt zusammengefasst. Grafisch ist der Workflow in Abbildung 3.3 dargestellt.

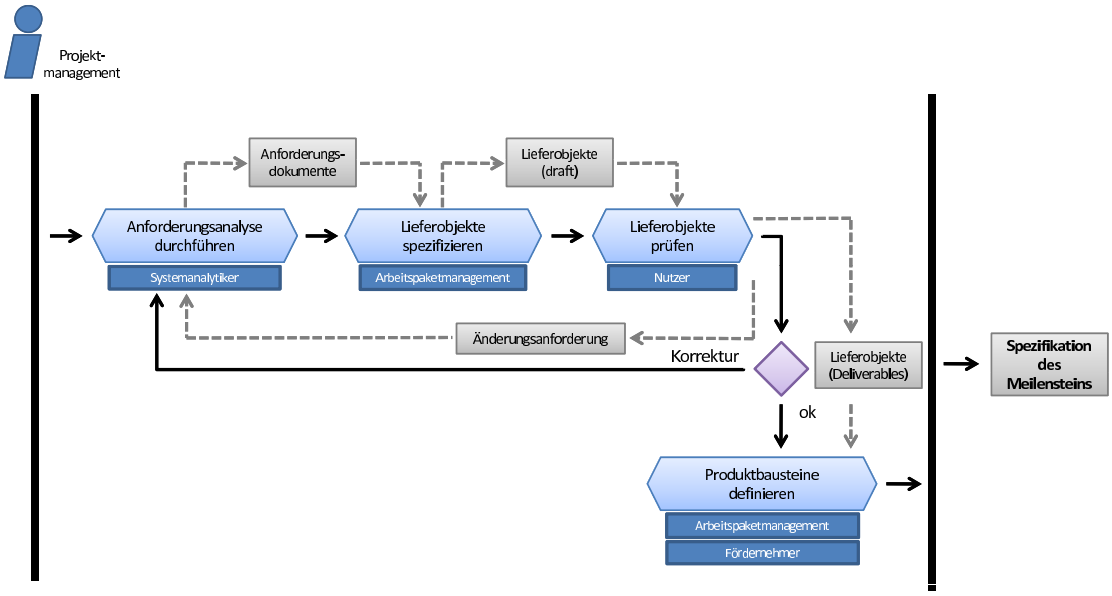

Abbildung 3.3: Workflow zur Spezifizierung eines Meilensteines als Vorbereitung für die Bearbeitung im Softwareevolutionsprozess.

Die Phase wird vom Projektmanagement initiiert und beginnt mit der Anforderungsanalyse, welche vom Systemanalytiker durchgeführt wird. Als Ergebnis entsteht ein Anforderungsdokument, das als Informationsinput für die folgende Aktivität der Paketmanager dient. Diese spezifizieren auf Basis des Anforderungsdokumentes Lieferobjekte, die während der späteren Bearbeitung des Meilensteins entstehen sollen. Als Ergebnis dieser Aktivität ergibt sich eine nicht endgültige Version der Beschreibung der Lieferobjekte, die an die Nutzer zur Prüfung weitergeleitet wird. Die Nutzer prüfen den Entwurf der Lieferobjekte und entscheiden, ob dieser korrigiert oder bestätigt wird. Im Falle einer Korrekturanforderung wird ein Schritt zurück in die Anforderungsanalyse gemacht, um diese erneut vom Systemanalytiker unter Einbeziehung der entstandenen Korrekturanforderungen überprüfen zu lassen. Von dort verläuft der Prozess wieder wie oben beschrieben. Im Falle der Bestätigung der Lieferobjekte wird deren Beschreibung an die Fördernehmer weitergegeben, die daraus unter Aufsicht des zuständigen Arbeitspaketmanagers Produktbausteine definieren, aus denen das entsprechende Lieferobjekt zusammengesetzt ist. Dies schließt die Definition des Meilensteins ab.

Meilenstein bearbeiten und fertigstellen

Nach der Spezifikation des Meilensteins sieht der Prozess die Umsetzung und Fertigstellung als nächsten Schritt vor. In dieser Phase, siehe Abbildung 3.4, wird die zuvor erstelle Spezifikation als

Grundlage für die Arbeit benutzt.

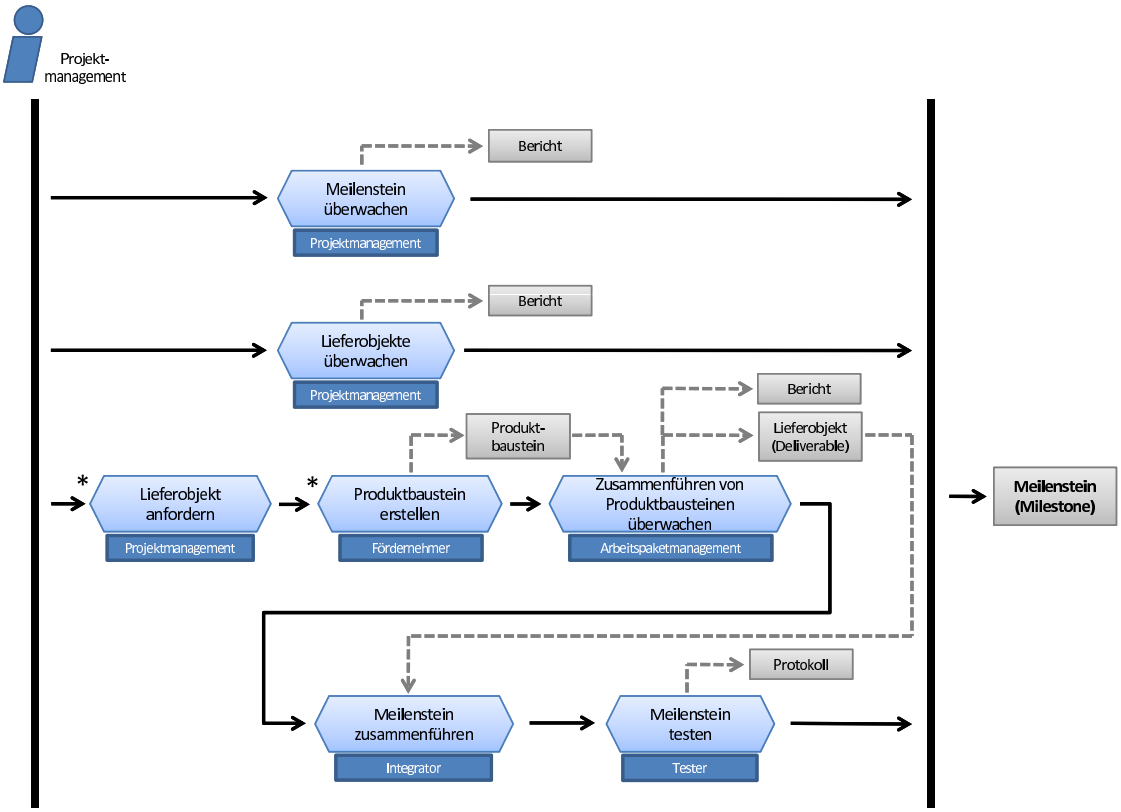

Abbildung 3.4: Workflow für die Bearbeitung und Fertigstellung eines Meilensteins im Softwareevolutionsprozess.

Auch diese Phase wird vom Projektmanagement eingeleitet. Es ist zusätzlich für die Überwachung des gesamten Meilensteins und der einzelnen Lieferobjekte zuständig. Dabei übernimmt das Projektmanagement nicht die Aufgaben des Arbeitspaketmanagers, sondern sorgt dafür, dass in der Meilensteinspezifikation festgelegte zeitliche Rahmenbedingungen für die Fertigstellung der einzelnen Lieferobjekte eingehalten werden, so dass eventuell existierende Abhängigkeiten zwischen diesen nicht zu unnötigen Verzögerungen führen. Die formalen Rahmenbedingungen von Grid-Projekten sehen ebenfalls vor, dass das Projektmanagement Berichte über den Fortgang und Status der Entwicklungsarbeiten anfertigt. Diese erscheinen als Artefakte im Prozess.

Parallel zur überwachenden Tätigkeit des Projektmanagements, ist es im Projektverlauf dafür verantwortlich die zu erstellenden Lieferobjekte entsprechend der Meilensteinspezifikation anzufordern. Dies initiiert die Bearbeitung der Produktbausteine durch die Fördernehmer. Diese Produktbausteine werden dann, überwacht vom Arbeitspaketmanagement, zu einem Lieferobjekt zusammengeführt.

Im Rahmen des Abschlusses der Phase werden die einzelnen Lieferobjekte von dem Integrator zu dem vollständigen Meilenstein zusammengeführt und die technische Funktionsfähigkeit vom Tester überprüft. Ein erfolgreicher Abschluss des Testes schließt ebenfalls die Fertigstellungsphase des Meilensteins ab. Feedbackmechanismen bei fehlgeschlagenem Test sind auf dieser Ebene der Prozessspezifikation der Übersichtlichkeit halber nicht angegeben.

Meilenstein veröffentlichen

Die Iteration des evolutionären Softwareentwicklungsprozesses wird durch die Veröffentlichung des Meilensteins abgeschlossen. Dabei durchläuft das Ergebnis der vorherigen Phase einen Verifikationsprozess, der zu eventuellen nachträglichen Korrekturen und dem abschließenden Release des Produktes führt, siehe Abbildung 3.5.

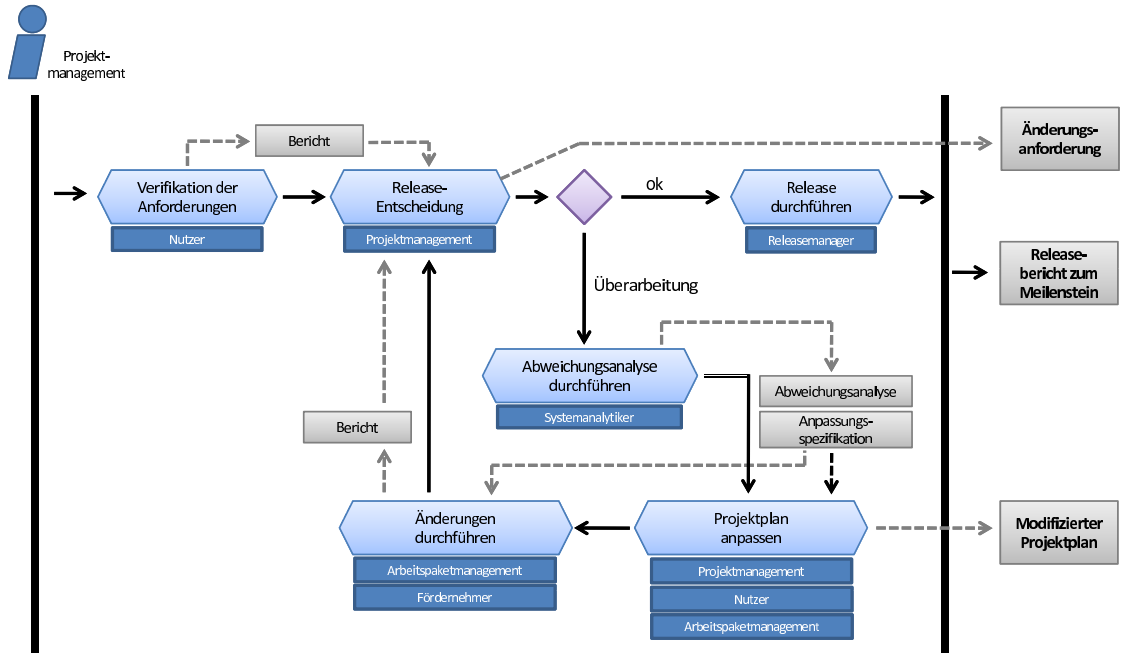

Abbildung 3.5: Workflow für das Release eines Meilensteins im Softwareevolutionsprozess.

Zu Beginn dieser Phase steht die Verifikation der Anforderung im Sinne der Benutzung der entwickelten Software durch die Nutzer. Sie fassen die Ergebnisse der Evaluation in einem Bericht zusammen, der dem Projektmanagement als Entscheidungsgrundlage für eine Releaseentscheidung dient. Fällt diese Entscheidung positiv aus, so übernimmt der Releasemanager die Veröffentlichung des Softwareproduktes als neue Generation und schließt damit die Meilensteinbearbeitung und Prozessiteration ab.

Im Falle einer negativen Releaseentscheidung liegen Abweichungen zur ursprünglichen Meilensteinspezifikation vor. Diese werden dann vom Systemanalytikter zusammengefasst und eine Anpassungsspezifikation geschrieben, die dem Projektmanagement als Grundlage für eine Anpassung des Projektplanes aufgrund der hinzugekommenen Änderungsnotwendigkeiten dient. Bei dieser Tätigkeit wird das Projektmanagement durch die Nutzer und Arbeitspaketmanager unterstützt. Die Nutzer können dabei zur Einschränkung der Ziele beitragen oder die Wichtigkeit von Anpassungen priorisieren. Die Arbeitspaketmanager tragen technische Aspekte bei, um eine genauere zeitliche Abschätzung im geänderten Projektplan aufstellen zu können.

Basierend auf der Abweichungsanalyse werden dann die Änderungen durch die Fördernehmer unter Aufsicht der Projektmanager durchgeführt. Die durchgeführten Änderungen werden in einem Bericht an das Projektmanagement dokumentiert und dienen nun als Grundlage für eine erneute Releaseentscheidung des Projektmanagements.

Literaturverzeichnis

[1] Fowler, M., The New Methodology. Online Publication , pp. 1 - 18, 13. Dez. 2005.

[2] Glinz, M., Software Engineering - Eine Einführung, Technischer Bericht, Universität Zürich, 2005.

[3] Kruchten, P., Der Rational Unified Process - Eine Einführung, Addison-Wesley-Longman, 1999.

[4] Larman, C., Basili, V. R., Iterative and Incremental Development: A Brief History. In IEEE Computer, pp. 47 - 56, June 2003, IEEE Press.

[5] Lehman, M. M., Programs, Life Cycles, and Laws of Software Evolution, Proceedings of the IEEE, 68(9), pp. 1060 - 1076, September 1980.

[6] Lehman, M. M., Laws of software evolution revisited, in Software Process Technology, LNCS 1149, pp. 108 - 124, Springer, 1996.

[7] Lehman, M. M., Ramil, J. F., Software evolution - Background , theory, practice, Information Processing Letters 88, pp. 33-44, Elsevier, 2008

[8] Versteegen, G., Software Management. Springer, Heidelberg, 2002.